Movie Store Inventory System Design Document

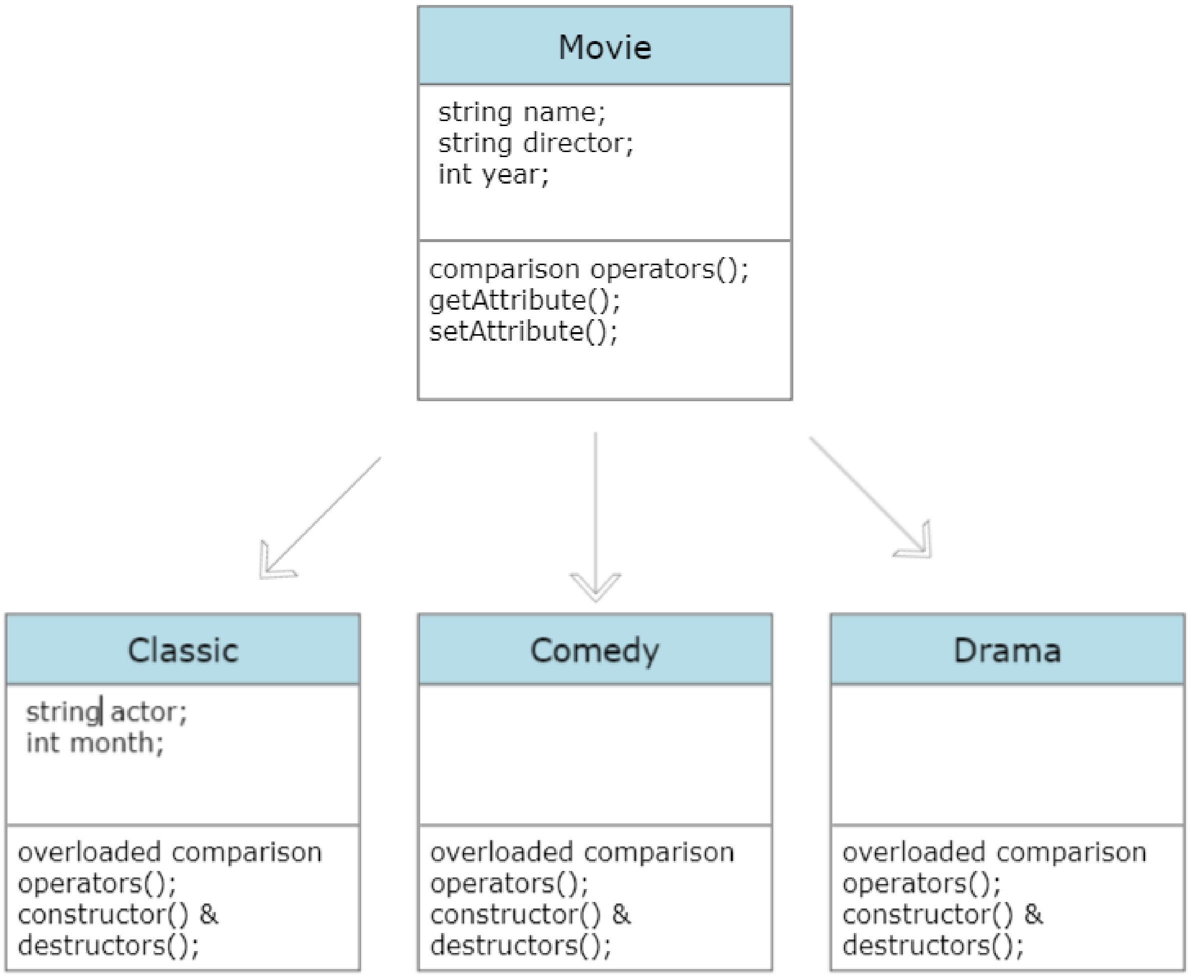
ASSIGNMENT 4 – CSS 343

KUZEY GOK

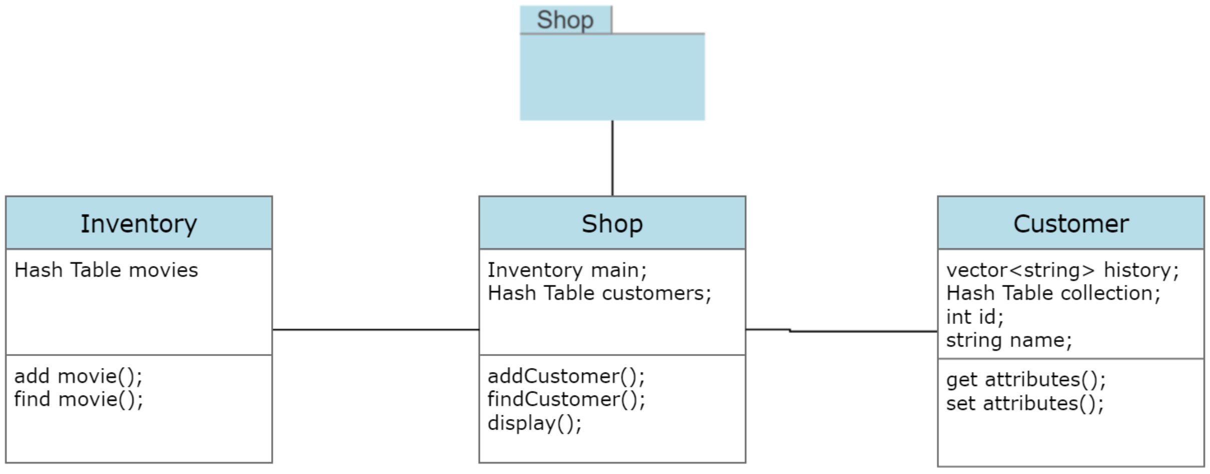
Overview

This system is designed for a movie rental store where inventory and movies are kept track of. In the main of this system, I use a parser to evaluate the 3 files passed in as arguments. There are 4 main categories of classes in this system. The first category is the movies. There are three types of movies that are extended from the parent movie class. The second category is the shop category. The shop is the main class that has a list of the customers and an inventory. The third category is the Hash classes. This includes a node class to be used for open hashing and data storage, along with a Hash Table class for a general hash table implementation. The final category is a parser class that will parse the input passed into the main.

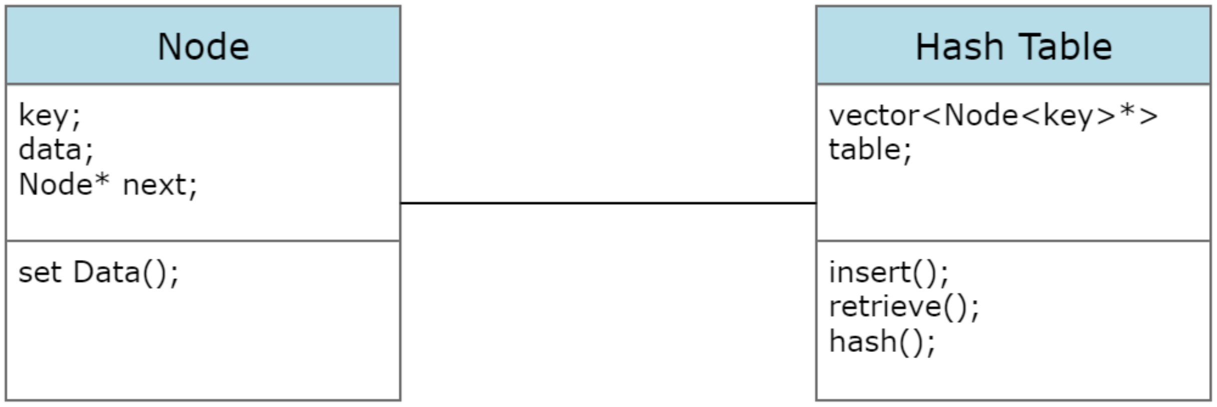
Movie Class Diagram


Movie
string name;
string director;
int year;
comparison operators();
getAttribute();
setAttribute();
Classic
string actor;
int month;
overloaded comparison operators();
constructor() & destructors();
Comedy
overloaded comparison operators();
constructor() & destructors();
Drama
overloaded comparison operators();
constructor() & destructors();

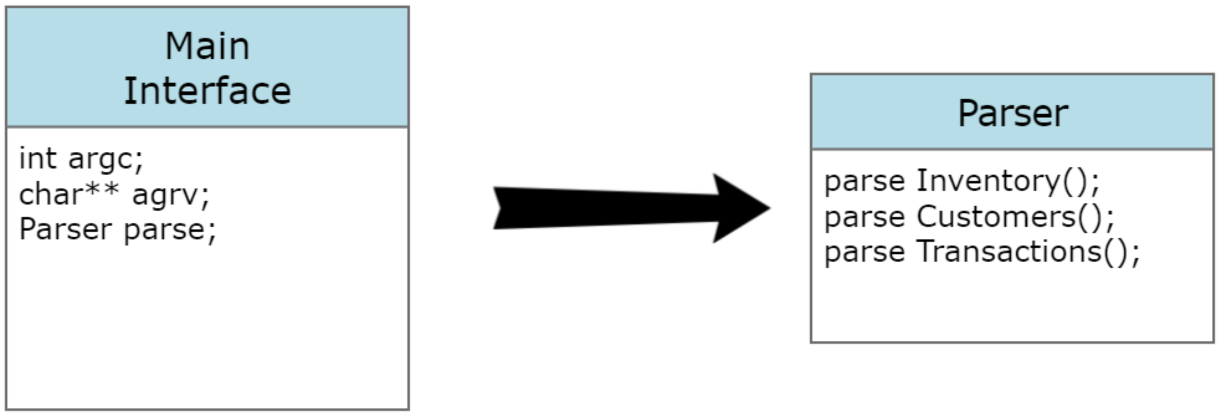
Shop Diagram



Hash Diagram

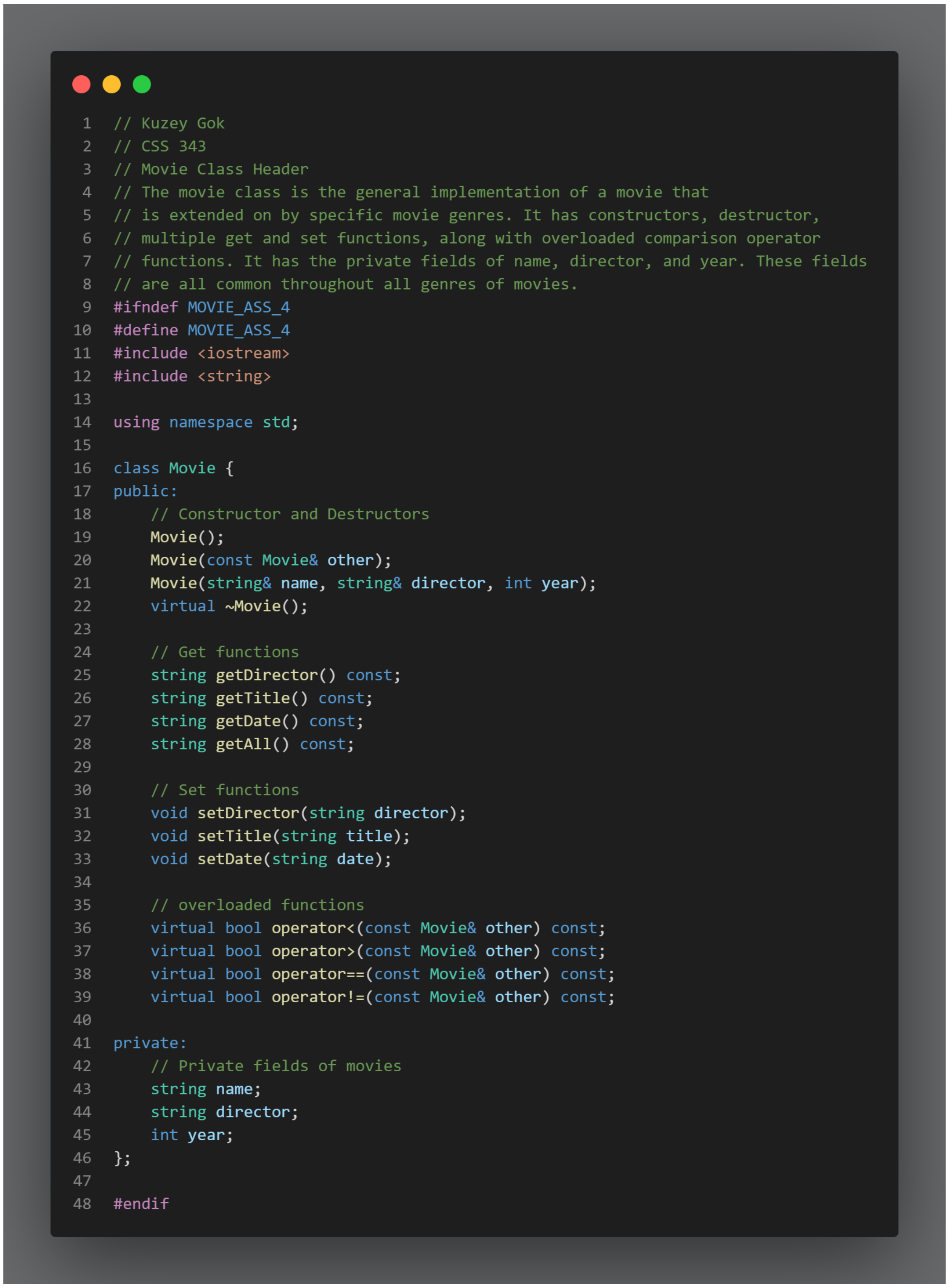


Parser Diagram



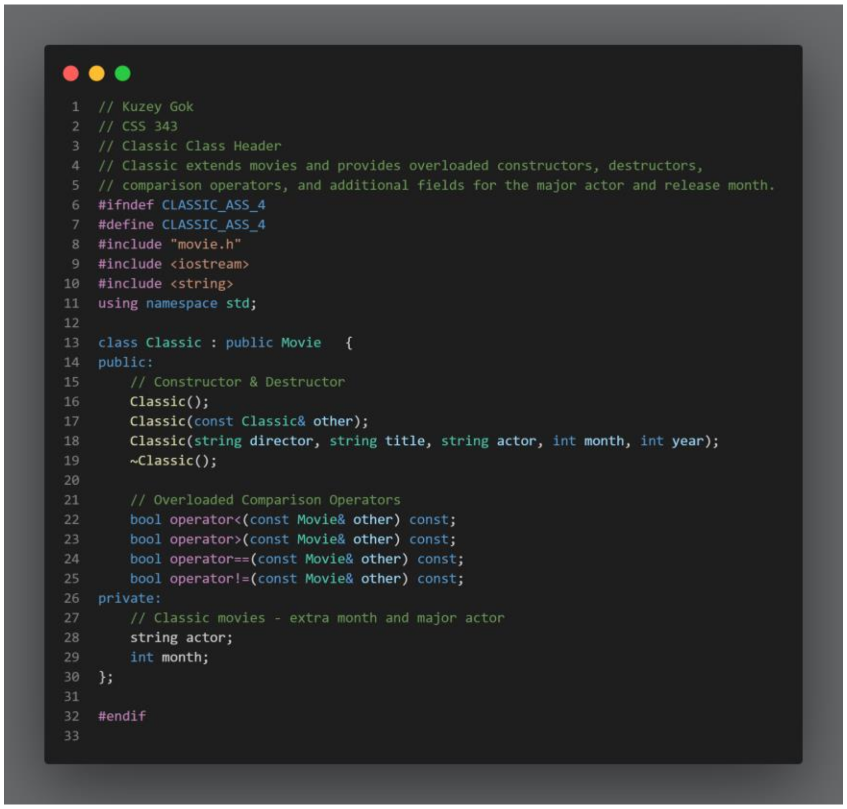
Class Descriptions

Movie:



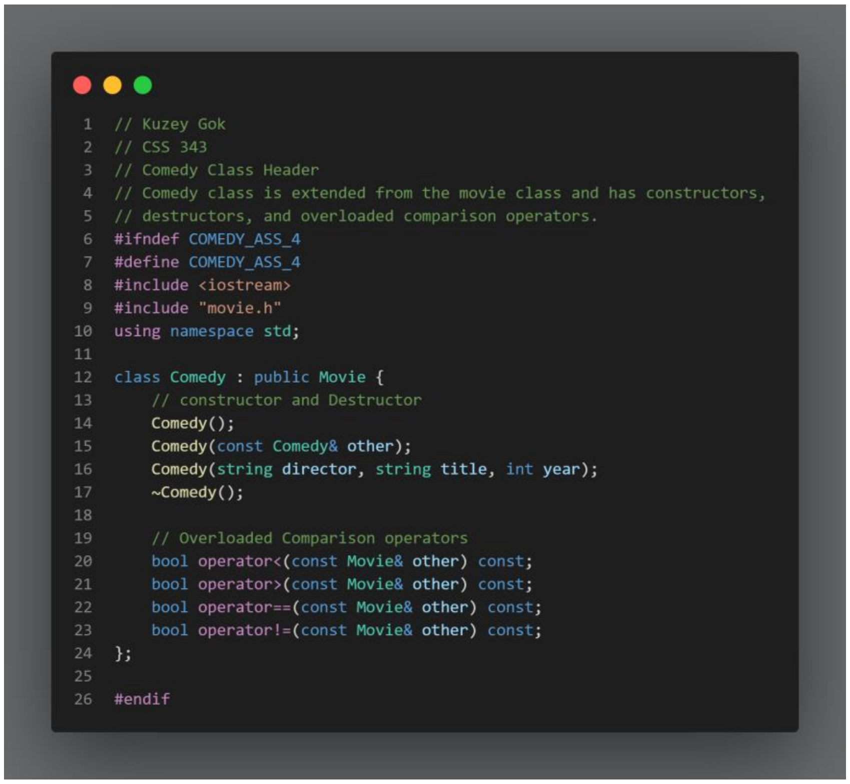
```cpp
// Kuzey Gok
// CSS 343
// Movie Class Header
// The movie class is the general implementation of a movie that
// is extended on by specific movie genres. It has constructors, destructor,
// multiple get and set functions, along with overloaded comparison operator
// functions. It has the private fields of name, director, and year. These fields
// are all common throughout all genres of movies.
#ifndef MOVIE_ASS_4
#define MOVIE_ASS_4
#include <iostream>
#include <string>

using namespace std;

class Movie {
public:
    // Constructor and Destructors
    Movie();
    Movie(const Movie& other);
    Movie(string& name, string& director, int year);
    virtual ~Movie();

    // Get functions
    string getDirector() const;
    string getTitle() const;
    string getDate() const;
    string getAll() const;

    // Set functions
    void setDirector(string director);
    void setTitle(string title);
    void setDate(string date);

    // overloaded functions
    virtual bool operator<(const Movie& other) const;
    virtual bool operator>(const Movie& other) const;
    virtual bool operator==(const Movie& other) const;
    virtual bool operator!=(const Movie& other) const;

private:
    // Private fields of movies
    string name;
    string director;
    int year;
};

#endif
```

Classic:



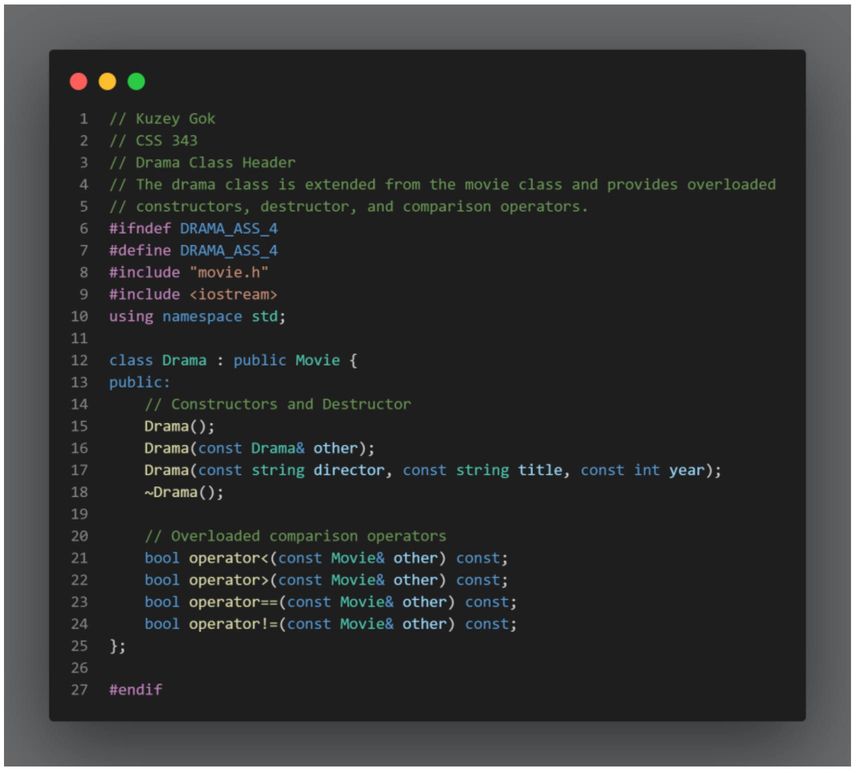
```cpp
// Kuzey Gok
// CSS 343
// Classic Class Header
// Classic extends movies and provides overloaded constructors, destructors,
// comparison operators, and additional fields for the major actor and release month.
#ifndef CLASSIC_ASS_4
#define CLASSIC_ASS_4
#include "movie.h"
#include <iostream>
#include <string>
using namespace std;

class Classic : public Movie   {
public:
    // Constructor & Destructor
    Classic();
    Classic(const Classic& other);
    Classic(string director, string title, string actor, int month, int year);
    ~Classic();

    // Overloaded Comparison Operators
    bool operator<(const Movie& other) const;
    bool operator>(const Movie& other) const;
    bool operator==(const Movie& other) const;
    bool operator!=(const Movie& other) const;
private:
    // Classic movies - extra month and major actor
    string actor;
    int month;
};

#endif

```

Comedy:



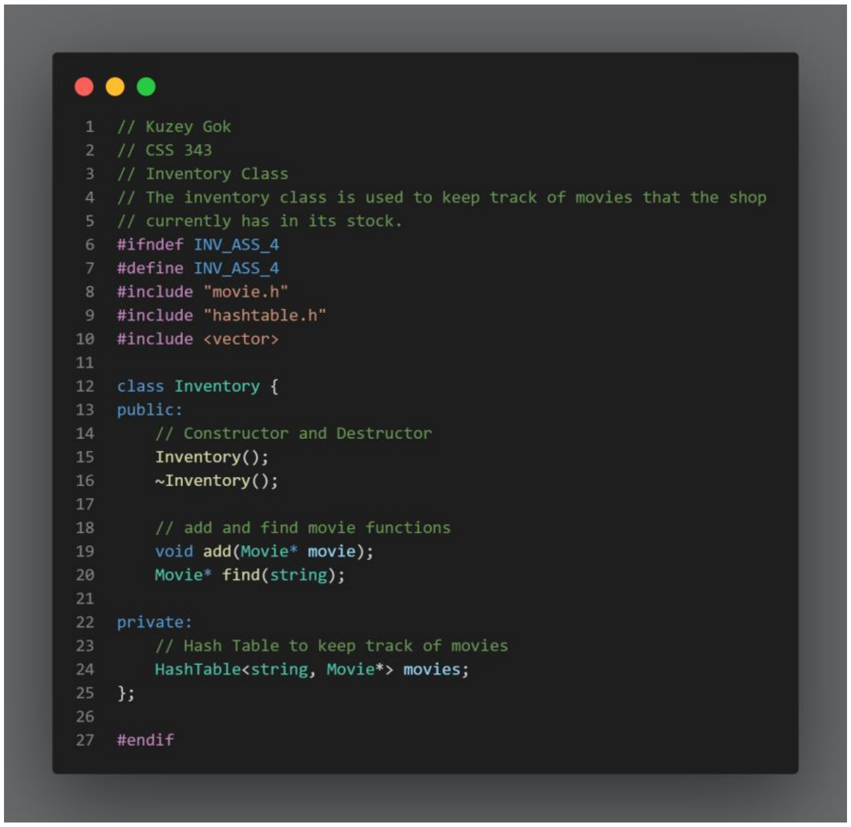
```cpp
// Kuzey Gok
// CSS 343
// Comedy Class Header
// Comedy class is extended from the movie class and has constructors,
// destructors, and overloaded comparison operators.
#ifndef COMEDY_ASS_4
#define COMEDY_ASS_4
#include <iostream>
#include "movie.h"
using namespace std;

class Comedy : public Movie {
    // constructor and Destructor
    Comedy();
    Comedy(const Comedy& other);
    Comedy(string director, string title, int year);
    ~Comedy();

    // Overloaded Comparison operators
    bool operator<(const Movie& other) const;
    bool operator>(const Movie& other) const;
    bool operator==(const Movie& other) const;
    bool operator!=(const Movie& other) const;
};

#endif
```

Drama:



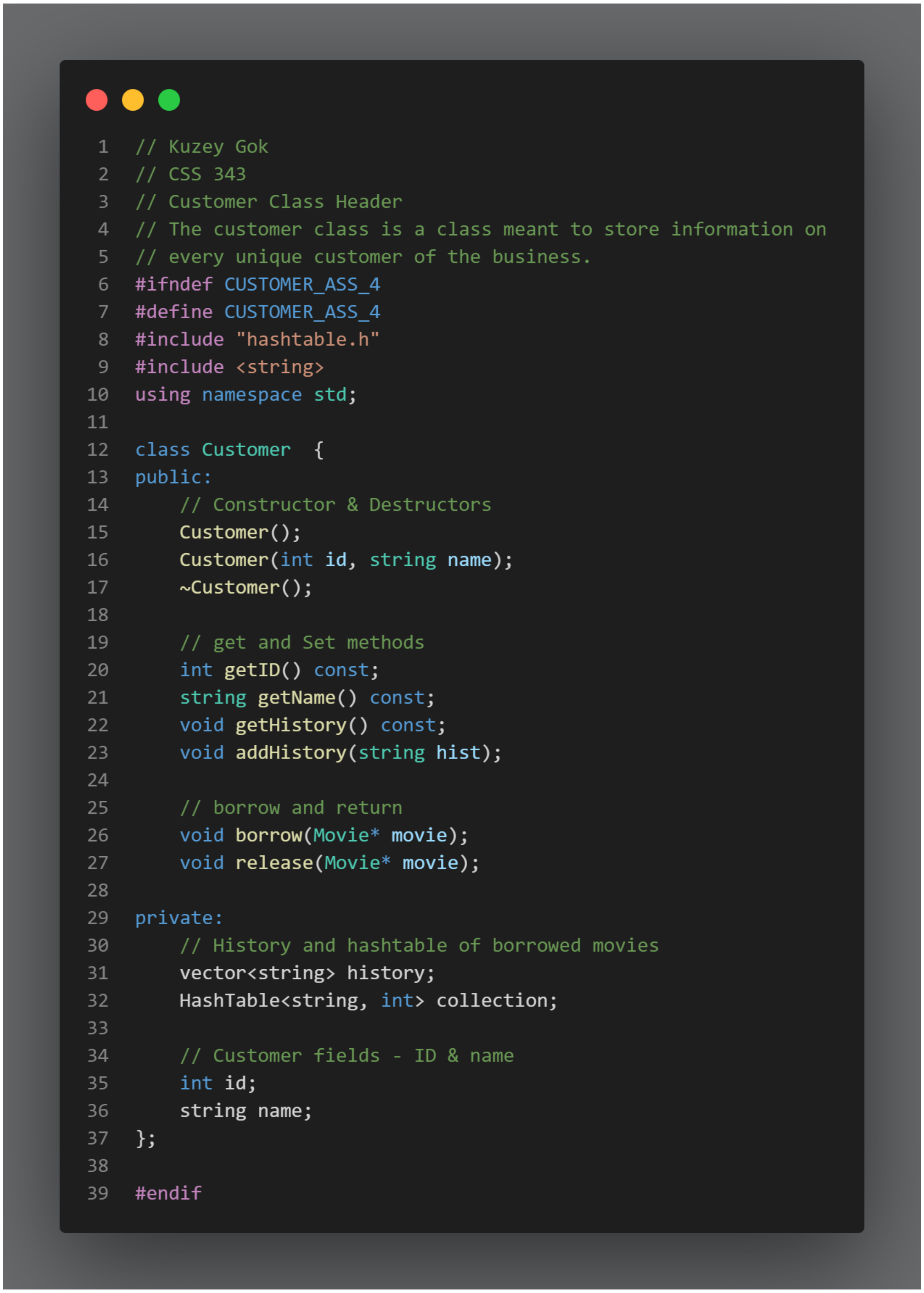
```cpp
// Kuzey Gok
// CSS 343
// Drama Class Header
// The drama class is extended from the movie class and provides overloaded
// constructors, destructor, and comparison operators.
#ifndef DRAMA_ASS_4
#define DRAMA_ASS_4
#include "movie.h"
#include <iostream>
using namespace std;

class Drama : public Movie {
public:
    // Constructors and Destructor
    Drama();
    Drama(const Drama& other);
    Drama(const string director, const string title, const int year);
    ~Drama();

    // Overloaded comparison operators
    bool operator<(const Movie& other) const;
    bool operator>(const Movie& other) const;
    bool operator==(const Movie& other) const;
    bool operator!=(const Movie& other) const;
};

#endif
```

Inventory:



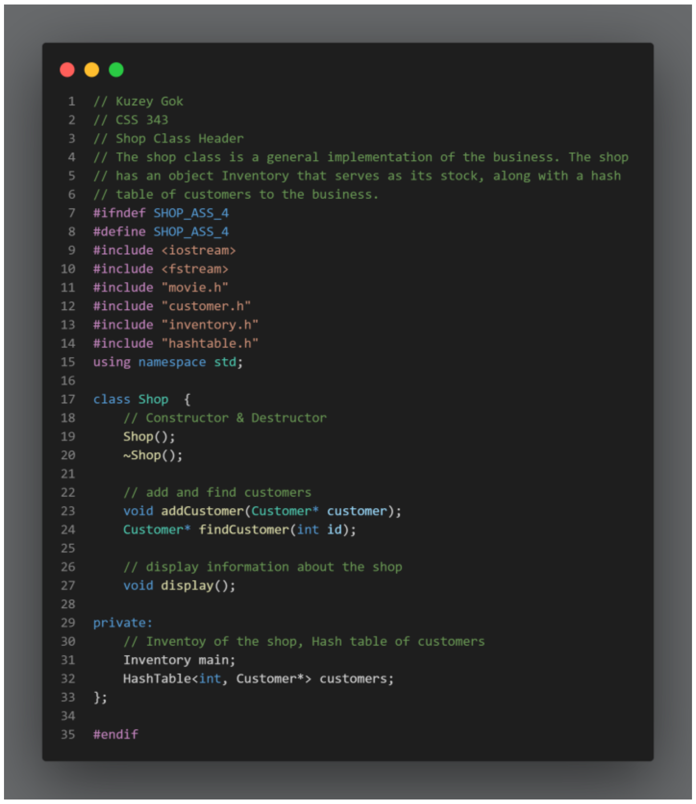
```cpp
// Kuzey Gok
// CSS 343
// Inventory Class
// The inventory class is used to keep track of movies that the shop
// currently has in its stock.
#ifndef INV_ASS_4
#define INV_ASS_4
#include "movie.h"
#include "hashtable.h"
#include <vector>

class Inventory {
public:
    // Constructor and Destructor
    Inventory();
    ~Inventory();

    // add and find movie functions
    void add(Movie* movie);
    Movie* find(string);

private:
    // Hash Table to keep track of movies
    HashTable<string, Movie*> movies;
};

#endif
```

Customer:



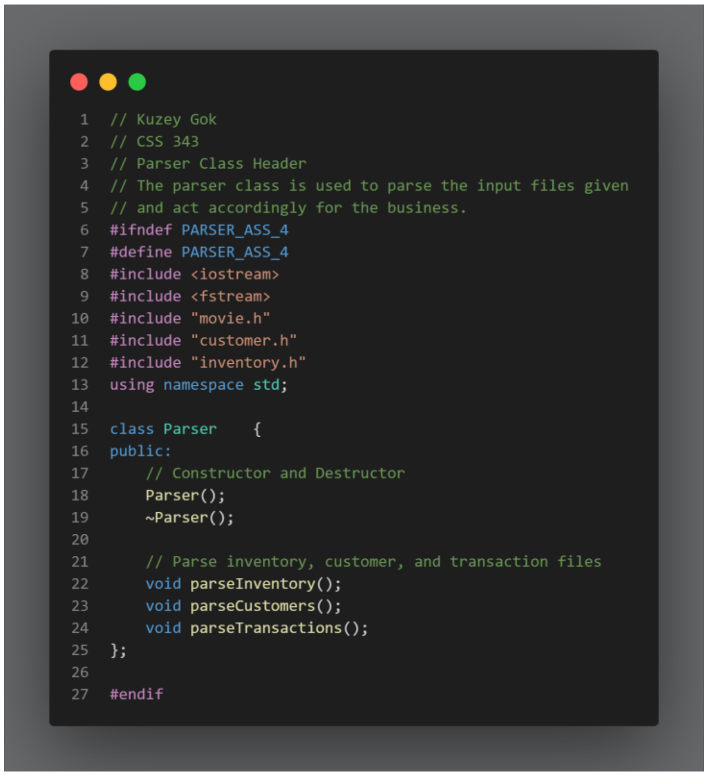
```cpp
// Kuzey Gok
// CSS 343
// Customer Class Header
// The customer class is a class meant to store information on
// every unique customer of the business.
#ifndef CUSTOMER_ASS_4
#define CUSTOMER_ASS_4
#include "hashtable.h"
#include <string>
using namespace std;

class Customer  {
public:
    // Constructor & Destructors
    Customer();
    Customer(int id, string name);
    ~Customer();

    // get and Set methods
    int getID() const;
    string getName() const;
    void getHistory() const;
    void addHistory(string hist);

    // borrow and return
    void borrow(Movie* movie);
    void release(Movie* movie);

private:
    // History and hashtable of borrowed movies
    vector<string> history;
    HashTable<string, int> collection;

    // Customer fields - ID & name
    int id;
    string name;
};

#endif
```

Shop:



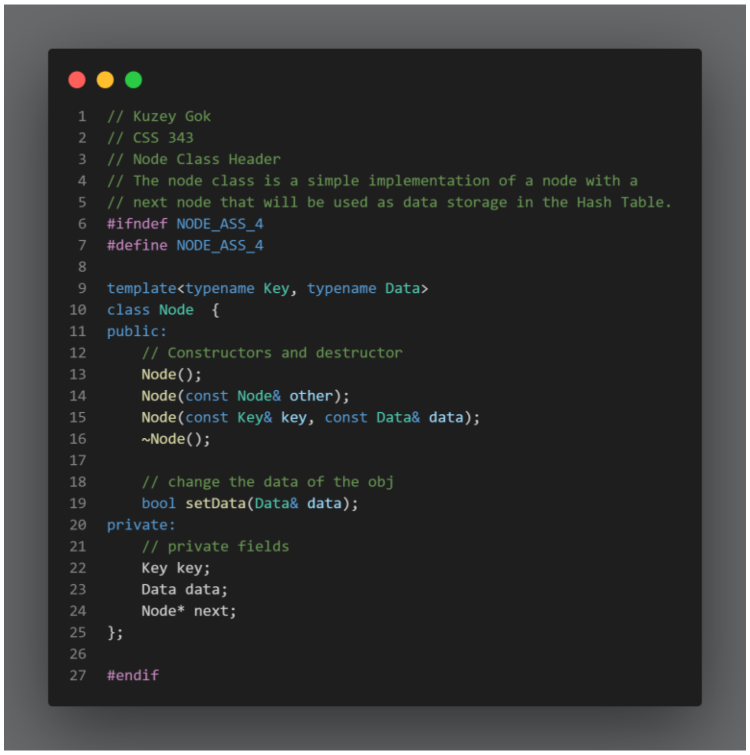
```cpp
// Kuzey Gok
// CSS 343
// Shop Class Header
// The shop class is a general implementation of the business. The shop
// has an object Inventory that serves as its stock, along with a hash
// table of customers to the business.
#ifndef SHOP_ASS_4
#define SHOP_ASS_4
#include <iostream>
#include <fstream>
#include "movie.h"
#include "customer.h"
#include "inventory.h"
#include "hashtable.h"
using namespace std;

class Shop  {
    // Constructor & Destructor
    Shop();
    ~Shop();

    // add and find customers
    void addCustomer(Customer* customer);
    Customer* findCustomer(int id);

    // display information about the shop
    void display();

private:
    // Inventoy of the shop, Hash table of customers
    Inventory main;
    HashTable<int, Customer*> customers;
};

#endif
```

Parser:



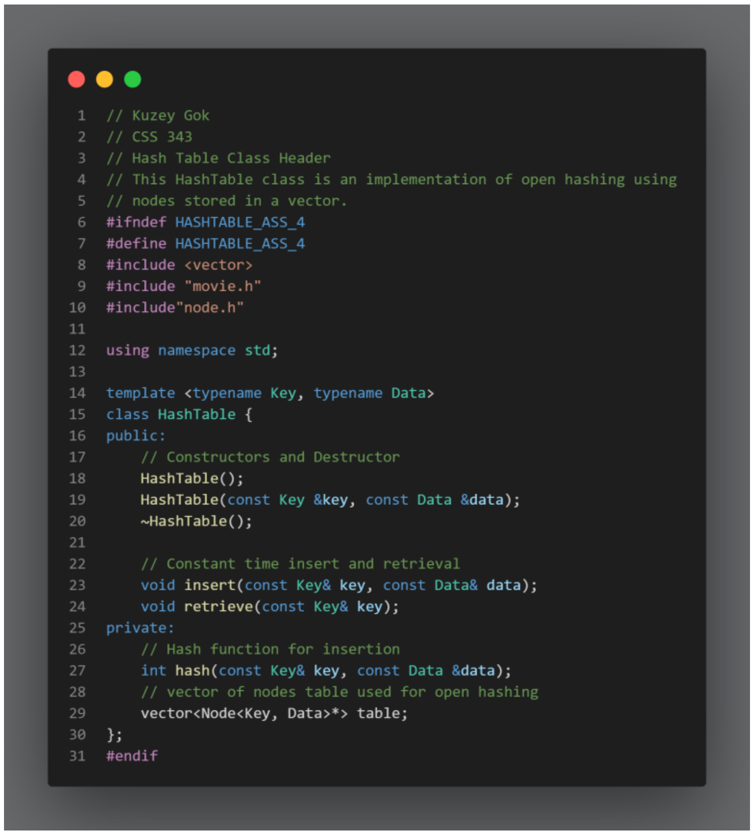
```cpp
// Kuzey Gok
// CSS 343
// Parser Class Header
// The parser class is used to parse the input files given
// and act accordingly for the business.
#ifndef PARSER_ASS_4
#define PARSER_ASS_4
#include <iostream>
#include <fstream>
#include "movie.h"
#include "customer.h"
#include "inventory.h"
using namespace std;

class Parser    {
public:
    // Constructor and Destructor
    Parser();
    ~Parser();

    // Parse inventory, customer, and transaction files
    void parseInventory();
    void parseCustomers();
    void parseTransactions();
};

#endif
```

Node:



```cpp
// Kuzey Gok
// CSS 343
// Node Class Header
// The node class is a simple implementation of a node with a
// next node that will be used as data storage in the Hash Table.
#ifndef NODE_ASS_4
#define NODE_ASS_4

template<typename Key, typename Data>
class Node  {
public:
    // Constructors and destructor
    Node();
    Node(const Node& other);
    Node(const Key& key, const Data& data);
    ~Node();

    // change the data of the obj
    bool setData(Data& data);
private:
    // private fields
    Key key;
    Data data;
    Node* next;
};

#endif
```

Hash Table:



```cpp
// Kuzey Gok
// CSS 343
// Hash Table Class Header
// This HashTable class is an implementation of open hashing using
// nodes stored in a vector.
#ifndef HASHTABLE_ASS_4
#define HASHTABLE_ASS_4
#include <vector>
#include "movie.h"
#include"node.h"

using namespace std;

template <typename Key, typename Data>
class HashTable {
public:
    // Constructors and Destructor
    HashTable();
    HashTable(const Key &key, const Data &data);
    ~HashTable();

    // Constant time insert and retrieval
    void insert(const Key& key, const Data& data);
    void retrieve(const Key& key);
private:
    // Hash function for insertion
    int hash(const Key& key, const Data &data);
    // vector of nodes table used for open hashing
    vector<Node<Key, Data>*> table;
};
#endif
```